JOURNAL OF SOFTWARE MAINTENANCE: RESEARCH AND PRACTICE
J. Softw. Maint: Res. Pract. **11**, 35–53 (1999)

Research

Experimental Evaluation of Agreement among Programmers in Applying the Rules of Cohesion

JAGADEESH NANDIGAM[1], ARUN LAKHOTIA[2]* and CLAUDE G. ČECH[3]

[1]*Department of Mathematics and Computer Science, Grambling State University, Grambling LA 71245, U.S.A.*
[2]*The Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette LA 70504, U.S.A.*
[3]*Psychology Department, University of Southwestern Louisiana, Lafayette LA 70504, U.S.A.*

SUMMARY

The cohesion or strength of a component of a software system is an indicator of its maintainability. The most popular way—as evidenced from coverage in textbooks—of determining the cohesion of a component is a set of rules developed by Stevens, Myers, Constantine and Yourdon in the early 1970s. Using Stevens *et al.*'s approach, a component is assigned one of seven levels of cohesion. This paper presents the results of an experiment analysing these rules of cohesion. The experiment, using fifteen computer science graduate students as subjects, was conducted to assess whether Stevens *et al.*'s rules were objective, i.e., whether there is a better-than-chance agreement in the cohesion levels assigned by different programmers. The data, though preliminary due to the small sample size, indicate that there is a significant variation in the cohesion levels assigned by them even though the subjects were assessed to have understood the concepts well. This decoupling between the understanding of the concepts of the scale and of the use of the scale in proper fashion is intriguing and deserves further study. The results also raise questions about the precision of the material taught in the software engineering curriculum. Copyright © 1999 John Wiley & Sons, Ltd.

KEY WORDS: software metrics; software measures; module cohesion; module strength; software engineering rules; levels of strength or cohesion

1. INTRODUCTION

There are two fundamental principles in engineering design: (a) maximize the functional relatedness of each component and (b) minimize the interdependencies between components. In the early 1970s, Stevens, Myers and Constantine (1974) observed that these principles were

*Correspondence to: Arun Lakhotia, The Center for Advanced Computer Studies, University of Southwestern Louisiana, PO Box 44330, Lafayette LA 70504–4330, U.S.A. Email: arun@cacs.usl.edu
Contract/grant sponsor: Army Research Office; Contract/grant number: ARO DAAH0494G-0334 and ARO DAAH049510250
Contract/grant sponsor: Army Research Labs; Contract/grant number: ARL DAAL019720159
Contract/grant sponsor: Louisiana Space Consortium; Contract/grant number: LaSPACE R-114223
Contract/grant sponsor: Louisiana Board of Regents; Contract/grant number: LEQSF 1991-92 ENH-98 and LEQSF 1993-95 RD-A-38

CCC 1040–550X/99/010035–19$17.50

Received 27 April 1998
Revised 13 November 1998

also relevant for designing software. Stevens *et al.* called the interconnections between modules (commonly called procedures) coupling and the functional relatedness of the activities performed by a module its cohesion or strength (Myers, 1978). Based on their interviews with software designers and on personal experiences, they identified several different types of relations of coupling and cohesion. They further rank-ordered each of the two sets of relations in decreasing order of preference of a designer. They ordered cohesion relations from high to low, and coupling from low to high. Stevens *et al.* claimed that system designs leading to modules with a higher level of cohesion and a lower level of coupling were qualitatively better.

This paper relates only to the notion of cohesion. The cohesion of a component of a software system is considered to be an indicator of its maintainability. Components with low cohesion are modified more often since they implement multiple functions. Such components are also more difficult to modify because a modification to one functionality may affect other functionalities. Thus, low cohesion implies lower maintainability. In contrast, components with high cohesion are modified less often and are also easier to modify. Thus, high cohesion implies higher maintainability.

Like several other concepts in software engineering, Stevens *et al*.'s cohesion (and coupling) relations have been accepted because of their intuitive appeal to software engineering researchers and professionals. These relations have so far not been subjected to any formal, systematic study. This is partly because Stevens *et al.* considered cohesion to be an attribute of design (Briand, Morasca and Basili, 1994). Thus, they and other subsequent authors have been content with assessing the cohesion of a module by analysing the natural language description of its purpose (Myers, 1978). As a result, it has been accepted that there is no obvious 'measurement procedure' for determining the level of cohesion in a module (Fenton, 1991, p. 200).

A question that has not been asked is: can Stevens *et al*.'s rules be used to determine the cohesion of a module from its source code? Or, more accurately: does significant agreement exist among programmers' assessment of the cohesion of modules arrived at by analysing the source code? This paper presents the results of a controlled experiment conducted to answer this question.

The above questions are important since Stevens *et al*.'s cohesion relations are paraphrased in several, if not most, software engineering textbooks (Jalote, 1990; Macro and Buxton, 1987; Pressman, 1992; Shooman, 1983). There have also been proposals for source code based measures for cohesion (Bieman and Ott, 1994; Boloix and Robillard, 1988; Emerson, 1984; Lakhotia, 1993; Nandigam, 1995). These proposals implicitly or explicitly map their measures of cohesion to the cohesion levels proposed by Stevens *et al.*, although such mappings have not been validated.

A positive answer to the question being investigated would have important ramifications towards the development and validation of source-code-based measures for cohesion. It would imply that the claim of mapping between a proposed measure of cohesion and Stevens *et al*.'s levels may be validated experimentally by comparing the cohesion levels (or numbers) assigned by the proposed measure to a module with the cohesion levels assigned to the module by subject programmers. A negative answer, on the other hand, may well suggest the need for further development of the concept of a scale of cohesion. If the cohesion levels assigned to a module by different programmers disagree, is that a sign of weakness in the software design? A negative answer to the question being investigated would raise concerns about the precision of at least one concept being taught in software engineering, an attribute that is clearly not desirable for a discipline to be qualified as engineering. Such a result would further amplify the need for experimental evaluation of software engineering concepts and principles (Glass, 1994; Fenton, Pfleeger and Glass, 1994).

The rest of this paper is organized as follows. Section 2 summarizes the notion of cohesion and the method for assessing the cohesion of a module as proposed by Stevens *et al.* Section 3 presents the design of our experiment. Section 4 tabulates the data collected, presents our experimental hypotheses and analyses the data to see if the null hypothesis can be rejected. Section 5 contains orthogonal data that may be used to test the validity of the experimental data and/or further support their conclusions. Section 6 concludes the paper by summarizing the results, their implications and the limitations of our experiment.

2. CLASSIFICATION OF COHESION

In the early 1970s Constantine attempted to learn why designers associated things into modules. He found that they used certain relationships between a set of actions to determine whether or not they should be performed in the same module. He termed these relationships associative principles (properties or characteristics) used by designers to associate actions to be placed in a module. He classified three such principles and arranged them in a linear order (or levels) reflecting the preferences of most designers for one principle over another. The ordinal scale defined by the set of associative principles along with their order he termed cohesion. Stevens, Myers, Constantine and Yourdon expanded the list of associative principles to seven (Stevens, Myers and Constantine, 1974; Yourdon and Constantine, 1978), which has now become the *de facto* standard and is paraphrased in most software engineering textbooks (Jalote, 1990; Macro and Buxton, 1987; Pressman, 1992; Shooman, 1983). Stevens *et al.*'s relations appear to be the most widely accepted method for measuring the cohesion (and coupling) of software components, and indirectly the maintainability of a system.

Table 1 enumerates the associative principles identified by Stevens *et al.* and their corresponding cohesion levels. The associative principles give the cohesion between pairs of processing elements. The cohesion of a module on the whole, as per Stevens *et al.*, is defined as the lowest of the cohesions between all pairs of processing elements. The informal definitions of the processing elements and of the associative principles make the concept of cohesion subjective.

Table 1. Associative principle between two processing elements and the corresponding cohesion in increasing order of levels, where the cohesion of a module is defined as the lowest cohesion between all pairs of its processing elements

Cohesion	Associative principles
Coincidental	None of the following associations hold
Logical	At each invocation, one of them is executed
Temporal	Both are executed within the same limited period of time during the execution of the system
Procedural	Both are elements of some iteration or decision operation
Communicational	Both reference the same input data set and/or produce the same output data set
Sequential	The output of one serves as input for the other
Functional	Both contribute to a single specific function

Table 2. Subjects' background information

Subject background variable	Low	Mean	High
Years of programming	2	4.6	11
Number of computer science courses as undergraduate	0	10	24
Number of computer science courses in Graduate school	3	8	15
Number of programming languages known	1	4	7

Table 3. Subjects' familiarity with the C language and cohesion concepts, on a scale of 0 to 10, where 0 = never used/heard, 5 = about average and 10 = expert

Subject background variable	Low	Mean	High
Familiarity with C language	5	7.6	9
Familiarity with cohesion concepts	3	6.1	8

3. EXPERIMENT DESIGN

3.1. Subjects

Section 3 presents the design of our experiment conducted to assess whether Stevens *et al.*'s cohesion relations are objective. It describes the background of the experimental subjects, the programs used for the analysis, material (such as questionnaires, etc.) used in the experiment and the procedure followed. The precise null hypothesis is stated in the next section. In this first subsection, we describe the background of the experimental subjects.

The experiment was conducted as part of a graduate level course on software engineering at the University of Southwestern Louisiana. Of the 16 students in this course, 15 volunteered to participate, and one refrained. Tables 2 and 3 summarize information about the student subjects' educational background and relevant programming experience. These data were collected at the beginning of the experiment, as described later.

3.2. Experimental programs

We chose four C programs of a collection of 26 used earlier by Goradia in experiments conducted for his Ph.D. research (Goradia, 1993), and made publicly accessible over the Internet. The four programs were selected based upon their modest size and domain of application. A domain criterion was used to exclude programs, such as matrix inversion, that required the knowledge of specialized algorithms. Only programs whose domain of application and/or algorithm were expected to be known to Computer Science graduate students, either by virtue of their training or social experiences, were selected.

Table 4 lists the programs chosen for our experiment and presents the size-related characteristics of these programs. The lines of code measure used in this paper is the count of the number of new-line characters.

Table 4. Programs used and their size measures

Program code	Program name	Number of lines	Number of functions	Average lines/function
P-1	Expression evaluation	83	5	16.6
P-2	Tax form	161	6	26.8
P-3	Accounting	245	6	40.8
P-4	Bank promotion	172	4	43.0

3.3. Experiment material

The materials used for the experiment may be classified into two categories: (a) material for the administrator of the experiment, (b) material for the subjects of the experiment. The material for the administrator of the experiment contained the following items:

(i) description of the experiment,
(ii) instructions on how to conduct the experiment,
(iii) informed consent forms used to obtain the consent of the subjects,
(iv) a copy of the textbook section on module cohesion used for the lecture on module cohesion,
(v) experiment packets to be given to the subjects,
(vi) copies of a quiz to assess the knowledge of the subjects, and
(vii) slips containing program and subject codes used for assignment of programs to subjects in a random manner.

The packet given to each subject of the experiment contained the following material:

(i) description of the experiment,
(ii) instructions to the subjects,
(iii) section on module cohesion from a text book (Page-Jones, 1988),
(iv) source-code listing of the experimental program assigned to the subject,
(v) a data sheet to record the subject's cohesion level assignment;
(vi) a remark form to collect the subject's comments about the experiment, and
(vii) a questionnaire to collect the subject's educational background.

3.4. Experiment procedure

The experiment was conducted in four stages. The activities of each stage are briefly described below.

Stage 1

During a regular lecture of the course in which the experiment was conducted, the instructor (Lakhotia, one of the authors) told the students about the experiment that was being planned and its significance to the software engineering community. He said that subjects were being sought

for the experiment, that the participation was voluntary, and that the students were encouraged to participate. He emphasized that the experiment was not intended to evaluate the participants but was actually attempting to evaluate the material that would be presented to them, and that their participation or non-participation would not influence their grades.

In a lecture about two weeks later, the administrator of the experiment (another volunteer graduate student) once again talked about the experiment and sought volunteers by distributing informed consent forms. 15 students volunteered as subjects and signed the informed consent form; one student refrained.

The experiment material was designed so that the names of the participants appeared only on the informed consent forms. All other material presented later was coded using subject codes (as described later); the correspondence between a subject and the subject code was never recorded. The students were informed of this scheme to gain their confidence that their privacy would indeed be maintained.

Stage 2

The administrator of the experiment (the same volunteer as above) presented a lecture on the concept of module cohesion. The lecture was based on Stevens *et al.*'s work. Experiment packets, containing the material mentioned earlier, were distributed to the subjects.

The experiment packets were prepared in the following manner. There were a total of 15 packets, four for three of the programs, and three for one of the programs used in the experiment. Each packet was labelled with a program code and a subject code. The program codes were P-1, P-2, P-3, and P-4 (see Table 4).

Randomness in assigning subjects to programs was achieved by having the subjects draw from a box the program code they were to analyse. The subjects were assigned a subject code based on their program code by noting the subject code on their experiment packet. The subject codes were formed as follows: S_{ij} indicating the jth subject assigned to the program P_i. Each subject looked at only one program. To ensure anonymity of the subjects, we did not associate subject names with their codes. A maximum of four subjects was assigned to any one program.

The subjects took the experiment packets home to study the programs assigned to them. They were given one week to complete their task. They were asked to complete and return items (v), (vi) and (vii) of their packet (see the listing in Section 3.3).

Stage 3

In the third stage, each subject studied the program received and assigned a cohesion level to each function based on the original definition of module cohesion by Stevens *et al*. The subjects recorded the cohesion level on the data sheet provided (item (v) in the list shown in Section 3.3). They also noted their comments about their task on the remark forms (item (vi)) and completed a questionnaire about their educational background (item (vii)).

Stage 4

In the fourth stage, the subjects brought, to a regular class lecture, the completed forms in envelopes provided to them. The experiment administrator conducted this session. Before turning

Table 5. Cohesion assignments for the Expression Evaluation program (P-1)

Function name	LOC	S_{11}	S_{12}	S_{13}	S_{14}
compute	23	logical	logical	communicational	logical
operand_value	4	functional	functional	functional	functional
get_token	7	sequential	sequential	functional	procedural
evaluate	22	functional	logical	procedural	functional
main	11	communicational	sequential	functional	procedural

Table 6. Cohesion assignments for the Tax Form program (P-2)

Function name	LOC	S_{21}	S_{22}	S_{23}
initialize	19	temporal	temporal	temporal
schedule_A	15	functional	functional	functional
figure_tax	33	functional	logical	functional
compute_tax	20	sequential	functional	sequential
valid_data	17	temporal	logical	functional
main	26	sequential	functional	sequential

in their packets, the subjects were given a quiz on module cohesion. This quiz was to be used to detect and eliminate data by those participants who did not demonstrate an understanding of the basic concepts of module cohesion. The intent of the quiz was not to judge or evaluate the individuals participating in the experiment, but rather to check the validity of the data obtained from the experiment. The subjects enclosed their completed quiz in the packet along with the material completed in Stage 3. The subjects then turned in these completed packets to the experiment administrator.

4. DATA ANALYSIS AND RESULTS

4.1. Subjects' responses

Tables 5 to 8 summarize the cohesion levels assigned to the functions in programs P-1 to P-4 by the experimental subjects. The columns labeled LOC show the sizes of the functions in terms of the lines of code.

4.2. Hypothesis

This experiment investigates whether Stevens *et al.*'s definition of module cohesion is objective by asking the question: do the subjects agree on the cohesion of a function in a program? This question may be stated in terms of the following experimental hypotheses:

> Null hypothesis (H_0): subjects are incapable of using Stevens *et al.*'s relations, and therefore their assignments of cohesion levels will be randomly distributed.

Table 7. Cohesion assignments for the Accounting program (P-3)

Function name	LOC	S_{31}	S_{32}	S_{33}	S_{34}
initialize	15	functional	temporal	coincidental	temporal
change_monthly	22	functional	functional	temporal	sequential
process_transaction	15	communicational	logical	sequential	communicational
process_end_of_month	22	functional	temporal	functional	temporal
process_report	29	sequential	logical	temporal	procedural
main	41	communicational	logical	coincidental	procedural

Table 8. Cohesion assignments for the Bank Promotion program (P-4)

Function name	LOC	S_{41}	S_{42}	S_{43}	S_{44}
assess_cashflow	31	communicational	logical	communicational	communicational
assess_account_status	19	functional	functional	functional	functional
recommended_account	22	functional	functional	functional	functional
main	57	procedural	sequential	functional	communicational

Alternative hypothesis (H_1): subjects display a better-than-chance consistency amongst themselves in assigning cohesion levels after learning about Stevens *et al*.'s relations.

We tested the above null hypothesis (H_0) using three statistical tests. First we used a nominal statistical test that views the seven levels of cohesion as simple categories, not as scalar values. Next we used an analysis of variance (ANOVA) test that treats the seven cohesion relations to be ordered, as proposed by Stevens *et al*. Next we performed an omnibus analysis of variance that compares the cohesion levels assigned by the subjects with that assigned by a software tool.

4.3. Analysis using a nominal statistic

We now present the results of analysing the data using the binomial test, a nominal non-parametric statistical test. For an experiment to qualify as a binomial experiment, it must have the following four properties (Mosteller, Rourke and Thomas, 1961): (a) there must be a fixed number of trials, (b) each trial must result in a success or failure, i.e., it is a binomial trial, (c) all trials must have identical probabilities of success, and (d) the trials must be independent of each other.

These properties were satisfied by our experiment, as follows. (a) For each program used in the experiment, the number of functions when viewed as number of trials was fixed. (b) Each test for agreement on cohesion level assigned to a function resulted in a success/match or a failure/no-match. (c) All trials had identical probability of success, i.e., the probability of successful match on cohesion level of one function did not affect the probability of successful match on cohesion level of another function. (d) Assignment of cohesion level to one function was independent of the assignment of cohesion level to another function. Thus, the trials were independent of each other. Tables 9 to 12 provide, for each experimental program, the percentage of agreement between each pair of subjects.

Table 9. Percentage of agreement amongst subjects for the Expression Evaluation program (P-1)

Program: expression evaluation (P-1)			
	S_{12}	S_{13}	S_{14}
S_{11}	60%	20%	60%
S_{12}		20%	40%
S_{13}			20%

Table 10. Percentage of agreement amongst subjects for the Tax Form program (P-2)

Program: tax form (P-2)		
	S_{22}	S_{23}
S_{21}	33%	83%
S_{22}		33%

Table 11. Percentage of agreement amongst subjects for the Accounting program (P-3)

Program: accounting (P-3)			
	S_{32}	S_{33}	S_{34}
S_{31}	17%	17%	17%
S_{32}		0%	33%
S_{33}			0%

Table 12. Percentage of agreement amongst subjects for the Bank Promotion program (P-4)

Program: bank promotion (P-4)			
	S_{42}	S_{43}	S_{44}
S_{41}	50%	75%	75%
S_{42}		50%	50%
S_{43}			75%

Assuming that each cohesion level is equally probable, the probability of assigning any particular level of cohesion—i.e., the probability of success—is $1/7$ or 0.143. The cumulative probabilities (p-value) of success of a binomial test resulting from this probability of success are given in Tables 13 to 16.

An entry with '**' in these tables represents an agreement with a 0.05 level of significance; an entry with a '*' represents an agreement with a 0.10 level of significance; other entries are not statistically significant. The significance level, denoted by α, of a test is the probability of rejecting the null hypothesis when it is true (a Type I error in a statistical test) (Newmark, 1992, p. 482). We can observe from Tables 13 to 16 that there is little agreement amongst subjects on the assignment of

Table 13. Cumulative binomial probabilities (p-values) for the Expression Evaluation program (P-1)

Program: expression evaluation (P-1)			
	S_{12}	S_{13}	S_{14}
S_{11}	0.023**	0.538	0.023**
S_{12}		0.538	0.152
S_{13}			0.538

Table 14. Cumulative binomial probabilities (p-values) for the Tax Form program (P-2)

Program: tax form (P-2)		
	S_{22}	S_{23}
S_{21}	0.207	0.000**
S_{22}		0.207

Table 15. Cumulative binomial probabilities (p-values) for the Accounting program (P-3)

Program: accounting (P-3)			
	S_{32}	S_{33}	S_{34}
S_{31}	0.604	0.604	0.604
S_{32}		1.000	0.207
S_{33}			1.000

Table 16. Cumulative binomial probabilities (p-values) for the Bank Promotion program (P-4)

Program: bank promotion (P-4)			
	S_{42}	S_{43}	S_{44}
S_{41}	0.101*	0.010**	0.010**
S_{42}		0.101*	0.101*
S_{43}			0.010**

cohesion to various functions. The only exception to the above observations is the case of program P-4. For this program, as shown in Table 16, there is a strong agreement among the subjects S_{41}, S_{43} and S_{44} with a 0.05 level of significance ($\alpha = 0.05$). As may be seen from Tables 8 and 12, these subjects generally agreed on three of the four functions.

There is no statistically significant evidence to reject the null hypothesis. We also cannot completely reject the alternative hypothesis because the analysis of Bank Promotion program (P-4) supports the alternative hypothesis with a minimum α of 0.1. The results of the binomial tests of the two hypotheses are therefore inconclusive.

Table 17. Analysis of variance for the Expression Evaluation program (P-1)

Source of variation	SS	df	MS	F
Between functions	38.7	4	9.675	4.21**
Within functions	34.5	15	2.300	
Between subjects	6.0	3	2.000	
Residual	28.5	12	2.375	

Table 18. Analysis of variance for the Tax Form program (P-2)

Source of variation	SS	df	MS	F
Between functions	36.0	5	7.20	2.7*
Within functions	32.0	12	2.67	
Between subjects	5.33	2	2.67	
Residual	26.67	10	2.67	

Table 19. Analysis of variance for the Accounting program (P-3)

Source of variation	SS	df	MS	F
Between functions	21.0	5	4.20	1.03
Within functions	73.5	18	4.08	
Between subjects	32.5	3	10.83	
Residual	41.0	15	2.73	

Table 20. Analysis of variance for the Bank Promotion program (P-4)

Source of variation	SS	df	MS	F
Between functions	21.188	3	7.063	7.21***
Within functions	11.751	12	0.98	
Between subjects	2.188	3	0.729	
Residual	9.563	9	1.063	

4.4. Analysis using interval statistical tests

In using the binomial test, we have analysed the data for a strict match on the categories of cohesion, with inconclusive results. An alternative to categorical testing is to investigate the extent to which subjects agree that some functions are more cohesive than others—i.e., the relative ordering of functions on cohesiveness. For example, we can take subjects' assignment of cohesion levels for a given program as representing values on a scale of cohesiveness, and check the extent to which a group of subjects reliably use such a scale. This may be analysed using an analysis of variance to examine the reliability of measurements (Winer, 1971, pp. 283–289).

For each of the four programs used in the experiment, Tables 17 to 20 show the results of the analysis of variance (ANOVA) test performed after transforming the subjects' cohesion level

Table 21. Theta and estimate of the reliability of the mean of the k subjects for experimental programs

Program name	Program code	θ	r_k
Expression evaluation	P-1	0.6614	0.7257
Tax form	P-2	0.4157	0.555
Accounting	P-3	−0.0212	−0.0929
Bank promotion	P-4	1.251	0.833

assignments into numbers ranging from 1 to 7, where 1 represents functional and 7 represents coincidental. An entry for the value of F marked with '***' represents reliability at the 0.01 level of significance ($\alpha = 0.01$); an entry with a '**' represents reliability at the 0.05 level of significance; an entry with '*' represents 0.10 level of significance; and other entries are not significant.

Based on ANOVA, the subjects exhibit some significant agreement on a scale of cohesion for programs P-1 and P-4 at the 0.05 or better level of significance, and for program P-2 at the 0.1 level of significance. Thus, we may reliably reject the null hypothesis that cohesion level assignments are random.

Table 21 shows the unbiased theta (θ) and the unbiased estimate of the reliability, r_k, of the mean of the k subjects cohesion values for each of the programs used in the experiment. These estimates are conservative, as the analysis of variance did not correct for end-anchor effects. However, corrected analyses revealed essentially the same patterns of results. It is worth noting that the analyses of variance reported above are conservative in that they are equivalent to analyses in which subjects are nested within functions. Analyses that take advantage of the potential systematicity in error variance contributed by a given subject across functions display the same essential pattern of results, however. In these latter analyses the residual mean square is used as the error term.

Finally, we address the question of the validity of subjects' cohesion assignments. The fact that subjects display systematicity in their ratings need not indicate that they are employing the same ordering as Stevens *et al.* To address this question, each function was also assigned a cohesion level using an automated tool developed by Nandigam (1995) based on an encoding of Stevens *et al*.'s rule proposed earlier (Lakhotia, 1993). The tool found only four levels of cohesion in the subject programs: functional, sequential, communicational and coincidental. To determine whether the assignments of cohesion levels our subjects gave corresponded to those of Stevens *et al.*, the levels assigned by the tool were treated as constituting the independent variable in an analysis of variance examining rated cohesion. A single rating of cohesion for each cohesion level rated by a subject was obtained by averaging that subject's rating for all functions at that level. The mean rating for each level constituted the dependent data in the analysis. In this analysis, 11 subjects contributed ratings for cohesion level 1 (functional); eight contributed ratings for cohesion level 2 (sequential); four contributed ratings for cohesion level 3 (communicational); and seven contributed to cohesion level 7 (coincidental). The resulting data were then subjected to an omnibus analysis of variance in which subjects were treated as being nested within cohesion level.

Table 22 presents the mean rated cohesiveness for these four levels, and Table 23 summarizes the analysis of variance on these data. The analysis reveals a significant difference in ratings as a function of cohesiveness at the $\alpha = 0.10$ level. As may be seen from Table 22, the data do exhibit an

Table 22. Mean difficulty rating for the four identified levels of cohesion

Level	1	2	3	7
Mean rating	1.07	1.5	1.73	1.10

Table 23. Omnibus analysis of variance of the mean difficulty ratings

Source of variation	SS	df	MS	F
Between	12.169663	3	4.056554	2.42
Within	43.7	26	1.674244	

orderly increase in the rated cohesion level with actual increases in level, with the exception of the coincidental level. Thus, while our subjects do not assign cohesion in a fashion perfectly consistent with Stevens *et al.*, there is some evidence that they exhibit some of the same sensitivity to cohesion as found with the Stevens *et al.* scale. At the same time, the actual mean ratings suggest that our subjects do not adopt the precise categories of the Stevens *et al.* scale, accounting for the mixed results obtained with the nominal statistics. The results suggest that the notion of levels of cohesion is one that subjects may use with some reliability and validity, although more work needs to be done exploring why the functional and the coincidental code were given equivalent ratings.

4.5. Power of the experiment

The power of a statistical test is defined as the probability of rejecting a null hypothesis such as our H_0 when it is false. This probability is equal to $1-\beta$, where β is the probability of a Type II error. A Type II error occurs when we fail to reject H_0 when H_0 is false. It is not possible to specify the exact value for the power of the statistical tests conducted in this experiment because our hypothesis H_1 is an inexact hypothesis. Given the approach adopted here, this was not of grave concern.

Nevertheless, some factors that affect the power of statistical tests in our experiment are outlined here. The power of a statistical test can be increased by avoiding Type II errors. If Type II errors are to be avoided, then a relatively large sample size or a large α value is required. As the sample size increases, the power of a statistical test increases. In our experiment, since the sample size is very small (at most four subjects per program), the power of the experiment is weak. As the alpha level becomes more stringent (goes from 0.05 to 0.01), the power decreases. This situation is mitigated in our experiment as we have used alpha levels of 0.05 or 0.10. In our experiment, the major contributor to the weak power is the small sample size. In our experiment, the programs used did not contain many functions (at most six functions per program) and the majority of these functions displayed functional cohesion, as measured by the tool. Therefore, the stimulus materials used did not contain a sufficiently large number of functions with a good distribution of cohesion levels to improve the power of the experiment. In summary, the power of an experiment such as ours could be increased by assigning more subjects per program, by employing a *within-subjects* design (as opposed to a *between-subjects* design), by using programs with a sufficiently large number of functions, and/or by using programs whose functions exhibit a good distribution of cohesion levels.

5. ANALYSIS OF SUBJECTS' KNOWLEDGE OF COHESION

5.1. Subjects' performance in the quiz

In the last stage of the experiment, each subject had completed a quiz and a questionnaire. These provided a means to evaluate the quality of the data collected in the experiment. This section begins with a summary and analyses the resulting feedback.

The quiz was designed to gauge whether the subjects had a minimal understanding of the concept of cohesion. Using Bloom's taxonomy of educational objectives, the minimal level of knowledge of a subject in an area constitutes knowing its terminology and facts (Bloom, 1969). The quiz was designed to test this minimal knowledge because this is sufficient to assess whether the students have grasped the description of each relation, as given by Stevens *et al.* The quiz contained the following types of question:

(a) one true/false question regarding the definition of module cohesion,
(b) one question in which a list of cohesion levels, in no particular order, had to be rearranged in increasing order of cohesion,
(c) seven fill-in-the-blank type questions where the definition of a cohesion level had to be related to the name of the cohesion level, and
(d) six small code fragments, commonly used in the literature on module cohesion, for which the subjects had to assign a cohesion level.

The subjects' answers in the first three components of the quiz were 95% correct, which shows that they understood the concepts of module cohesion. Their answers in the last component (assignment of cohesion to code fragments) had some variations. This again shows that the original definitions are subjective in nature and difficult to apply to determine precisely the cohesion of a code fragment.

5.2. Feedback from the subjects

The subjects were also given a form to provide remarks about the experiment. The form contained a set of questions, given below, and space to provide descriptive feedback using one of the following four responses: easy, average, difficult or very difficult.

(1) How did you find the lecture on module cohesion to understand?
(2) How did you find the material on module cohesion to read and understand?
(3) How did you find the definitions of cohesion levels to understand?
(4) How did you find the program assigned to you to understand?
(5) How did you find the task of assigning of cohesion level to functions?

Table 24 summarizes the responses to the above questions. Each cell in this table gives a count. For each question, the table gives the distribution of the subjects' responses in the four levels of difficulty: easy, average, difficult, and very difficult.

Notice that none of the subjects felt that the material on module cohesion was difficult to understand (question 2). 86.6% of the subjects felt that the definitions of cohesions were not difficult

Table 24. Summary of feedback information from subjects, where each row shows the distribution of responses across various levels of difficulty

Question number	Easy	Average	Difficult	Very difficult
1	4	7	3	1
2	8	7		
3	8	5	2	
4	10	4	1	
5	3	8	4	

to understand—i.e., were either easy or average (question 3). 93.3% of the subjects said that the sample programs were not difficult, yet only 73.3% reported that the task of assigning cohesion levels to functions in those samples (question 5) was not difficult.

These responses would indicate that, by and large, the students understood the programs and the cohesion material well. Yet there is a significant variation in their assignment of cohesion levels to the subject programs. One reason for this variation may be that the original definitions of module cohesion by Stevens *et al.* are very intuitive, and hence are easy to understand. If so, the definitions can be interpreted differently, and thus lead to the significant variation seen.

The subjects' descriptive comments support this position, as exampled in these quotations:

1. 'The definition of levels of cohesion are intuitive. Therefore, assigning a level of cohesion to a function is also intuitive. It is hard to do objectively.'
2. 'The first three levels of cohesion (functional, sequential, communicational) were straightforward and easy to understand. The remaining levels of cohesion were confusing.' (This is consistent with the results of the omnibus analysis reported in Section 4.4.)
3. 'It is not very easy to give an accurate cohesion level assignment for the given source code.'
4. 'Making the final choice of cohesion level for each function is only as good as my best estimate.'

6. CONCLUSIONS

The maintainability of a software system may be assessed by the cohesiveness of its modules. Modules with high cohesion are easier to modify than modules with low cohesion. A programmer's ability to assess the cohesion of a module can influence that programmer's ability to develop maintainable programs and maintain them.

This paper reports the results of a preliminary study conducted to assess whether Stevens *et al.*'s rules of cohesion are objective. The question is significant since these rules are paraphrased in most software engineering textbooks (Jalote, 1990; Macro and Buxton, 1987; Pressman, 1992; Shooman, 1983). A study of this nature helps to evaluate the precision of material taught in the software engineering curriculum. Our own interest in this study was kindled by Lakhotia and Nandigam's measure for cohesion (Lakhotia, 1993; Nandigam, 1995) that has been developed by simply translating into logic the rules of cohesion proposed by Stevens *et al*. Since such a translation, to the extent possible, preserves the intent of the original rules, Lakhotia and Nandigam's measure

could be treated as a reference measure for comparing other proposed measures (Bieman and Ott, 1994; Boloix and Robillard, 1988; Emerson, 1984; Ott and Thuss, 1989, 1991). This experiment is a step in that direction.

Since the sample size of our experiment is small (15 students and four programs) the experiment lacked sufficient power to be conclusive. Nonetheless the data collected do provide us with evidence to draw preliminary conclusions, summarized below. Follow-up studies are needed to allay the issues raised.

1. *Is the cohesion scale so intuitively obvious that it can be used reliably?* The first series of analyses using binomial tests suggests that the answer is no. It may be the case that these tests were not sufficiently powerful, given the small number of students. But, as the students did exhibit comprehension of the concepts on the questionnaire, the results do not give aid or comfort to instructors teaching these concepts.
2. *If the cohesion scale is not used reliably, is it nevertheless the case that it reflects an important underlying idea that software modules can be rated in terms of relative cohesion?* According to our ANOVA analyses, although people do not agree on the categories, they apparently do share some notion that module A is less cohesive than module B. This result does not directly support the Stevens *et al.* scale. Students with intuitions directly opposing those from the Stevens *et al.* scale would have provided data yielding significant F in these analyses.
3. *If subjects reliably order code segments in terms of relative cohesion, is that ordering anything like the ordering the Stevens et al. scale would yield?* This can be studied by using Lakhotia and Nandigam's instrument as a substitute for the Stevens *et al.* scale. The average ratings suggest that the subjects were perhaps biased towards the low end of the scale, since levels 2 and up appear underestimated in the group averages. Still, the fact of a positive correlation in instrument and group ratings for the first three levels does fit the claim that these represent a proper relative ordering. At the same time, the average for the highest level places does not differ greatly from the lowest level, and that illustrates a clear problem in the Stevens *et al.* scale: how can people confuse the highest level of cohesion with the lowest? May it be an indicator of a major conceptual difficulty?
4. *Did the quality of the lecture affect the subjects' responses?* As shown in Table 24, 27% of the subjects, 4 out of 15, rated the lecture as difficult or very difficult. Did this cause interference? This can be studied by assigning the numerical ordering 1, 2, 3 and 4 to the categories from easy to very difficult, respectively. The mean difficulty rating for questions 1 to 5 are then, respectively 2.07, 1.47, 1.6, 1.4 and 2.07. This makes the mean difficulty rating for the lecture (question 1) 2.07, which corresponds to the average category, the same as for the task of assigning cohesion levels (question 5). Not only are the means different, but the shapes of the distributions are also different, making the question 1 response pattern significantly different from the patterns for questions 2 (0.01 level), 3 (0.02 level) and 4 (0.01 level), but not significantly different from question 5 (0.10 level). Supporting the position that the lecture may not have interfered with the results is that all of the subjects found the written material on cohesion to be average or easy to understand (question 2), and that the subjects had the written material when evaluating the subject programs. Further supporting the presumption that having the written material may have countered any negative influence of the lecture is that all of the subjects performed satisfactorily on the simple minimal-knowledge quiz given in Stage 4 of the experiment.

5. *Does the concept of cohesion lack sufficient precision to bridge the gap between the conceptual knowledge and procedural knowledge?* As already mentioned, all of the subjects found the written material of easy or average difficulty to understand, for an average rating of 1.47 for question 2. Furthermore, none of the subjects found the definitions of the various levels of cohesion very difficult to understand, for an average rating of 1.6 for question 3. If the definitions were understandable and the programs were similarly understandable (an average rating of 1.4 for question 4), then it follows that the subjects presumably felt reasonably certain that they knew the material at a conceptual level. But, knowing something at a conceptual level and being able to apply it are two different things. The answers to question 5 clearly show this. Despite an average rating of 2.07 for this question, 27% of the subjects rated the task of cohesion assignment as difficult. This reinforces the distinction between conceptual and procedural or application knowledge. The concept of cohesion appears to be one such concept that is relatively easy to understand at a conceptual level but is difficult to apply.
6. *If the cohesion scale is not intuitively obvious, can people at least be trained to use it reliably?* At present we do not have any data on this issue. The results concerning the relative agreements of instrument and ratings for the first three levels coupled with the commentary about the first three levels being relatively easy, at least holds out the promise that students might be taught to anchor their judgements to the scale categories. But in any case, the current results suggest that the type of introduction students receive, as again assessed by the questionnaires, provides knowledge of the relevant concepts, but does not necessarily provide sufficient knowledge of how to use these concepts in realistic settings. At that level, we can at least conclude that the Stevens *et al.* scale is not an intuitively easy scale to grasp, and that further work is justified in asking whether the scale has validity, or whether we might not be better off with some other measure of cohesion. The latter may be motivated by considering the results for level 7. Maybe we do not have a single-dimension concept, or maybe the notion of level 7 cohesion is too ill-defined at the moment.

Acknowledgements

The authors thank Anurag Bhatnagar and John Gravley for their help in conducting the experiments; Robert McFatter for his advise on experimental analysis; and the 15 students whose participation in the experiments made this study possible.

References

Bieman, J. M. and Ott, L. M. (1994) 'Measuring functional cohesion', *IEEE Transactions of Software Engineering*, **20**(8), 644–657.

Bloom, B. (Ed) (1969) *Taxonomy of Educational Objectives: The Classification of Educational Goals—Handbook 1: Cognitive Domain*, David McKay Co., Inc., New York NY, 207 pp.

Boloix, G. and Robillard, P. N. (1988) 'Interconnectivity metric for software complexity', *INFOR*, **26**(1), 17–39.

Briand, L., Morasca, S. and Basili, V. R. (1994) *Defining and validating high-level design metrics*, Technical Report CS-TR-3301, Computer Science Department, University of Maryland, College Park MD, 32 pp.

Emerson, T. J. (1984) 'A discriminant metric for module cohesion', in *Proceedings 7th International Conference on Software Engineering*, IEEE Computer Society Press, Los Alamitos CA, pp. 294–303.

Fenton, N., Pfleeger, S. L. and Glass, R. L. (1994) 'Science and substance: a challenge to software engineers', *IEEE Software*, **11**(4), 86–95.
Fenton, N. E. (1991) *Software Metrics: A Rigorous Approach*, Chapman & Hall, London, 337 pp.
Glass, R. L. (1994) 'The software research crisis', *IEEE Software*, **11**(6), 42–47.
Goradia, T. S. (1993) *Dynamic impact analysis: analyzing error propagation in program executions*, Ph.D. dissertation, Department of Computer Science, New York University, New York NY, 221 pp.
Jalote, P. (1990) *An Integrated Approach to Software Engineering*, Springer-Verlag New York, Inc., New York NY, 374 pp.
Lakhotia, A. (1993) 'Rule-based approach to computing module cohesion', in *Proceedings 15th International Conference on Software Engineering*, IEEE Computer Society Press, Los Alamitos CA, pp. 35–44.
Macro, A. and Buxton, J. (1987) *The Craft of Software Engineering*, Addison-Wesley Publishing Co., Reading MA, 380 pp.
Mosteller, F., Rourke, R. E. K. and Thomas, G. B. (1961) *Probability and Statistics*, Addison-Wesley Publishing Co., Reading MA, 395 pp.
Myers, G. J. (1978) *Composite/Structured Design*, Van Nostrand Reinhold Company, New York NY, 174 pp.
Nandigam, J. (1995) *A measure for module cohesion*, Ph.D. dissertation, The Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette LA, 156 pp.
Newmark, J. (1992) *Statistics and Probability in Modern Life*, 5th edn., Sanders College Publishing, New York NY, 739 pp.
Ott, L. M. and Thuss, J. J. (1989) 'The relationship between slices and module cohesion', in *Proceedings 11th International Conference on Software Engineering*, IEEE Computer Society Press, Los Alamitos CA, pp. 198–204.
Ott, L. M. and Thuss, J. J. (1991) *Slice based metrics for estimating cohesion*, Technical Report CS-TR-91-04, Department of Computer Science, Michigan Technological University, Houghton MI, 17 pp.
Page-Jones, M. (1988) *The Practical Guide to Structured Systems Design*, Prentice-Hall, Englewood Cliffs NJ, 368 pp.
Pressman, R. S. (1992) *Software Engineering: A Practitioner's Approach*, 3rd edn., McGraw Hill Book Co., New York NY, 793 pp.
Shooman, M. L. (1983) *Software Engineering: Design, Reliability, and Management*, McGraw-Hill Book Co., New York NY, 683 pp.
Stevens, W. P., Myers, G. J. and Constantine, L. L. (1974) 'Structured design', *IBM Systems Journal*, **13**(2), 115–139.
Winer, B. J. (1971) *Statistical Principles in Experimental Design*, 2nd edn., McGraw-Hill Book Co., New York NY, 907 pp.
Yourdon, E. and Constantine, L. L. (1978) *Structured Design*, Prentice-Hall, Englewood Cliffs NJ, 446 pp.

Authors' biographies:

Jagadeesh Nandigam is an Assistant Professor of Computer Science in the Department of Mathematics and Computer Science at Grambling State University, Grambling, Louisiana. He received his Ph.D. from the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, in 1995. His dissertation work was in the area of measures for module cohesion. His research interests include object-oriented software metrics, reverse engineering, program flow analysis and intelligent agents. His research projects are funded by grants from ARO, ARL, LaSPACE (Louisiana Space Consortium) and the Louisiana Board of Regents. Jagadeesh is a member of the ACM. His email address is: jagadeesh@alpha0.gram.edu

Arun Lakhotia is an Associate Professor with the Center for Advanced Computer Studies and the Director of Software Research Laboratory at the University of Southwestern Louisiana, Lafayette. He received his Ph.D. from Case Western Reserve University, Cleveland, in 1989. His research interests are centred around issues related to improving the productivity of programmers and reliability of their programs. He is currently directing projects to recover object-oriented design from source code of legacy systems. The projects are funded by grants from ARPA and the Louisiana Board of Regents. He has also contributed to research in methods and tools for developing logic programs, partial evaluation of logic programs, interprocedural flow analyses and software metrics. Arun is a member of the ACM and IEEE. His email address is: arun@cacs.usl.edu

Claude G. Čech holds appointments at the University of Southwestern Louisiana as a Professor in the Department of Psychology and an Adjunct Professor in the Center for Advanced Computer Studies. He obtained his Ph.D. in Cognitive Psychology from the University of Illinois in 1981. Claude's primary research interests include information processing models of mental comparison and magnitude representation, situational frequency sensitivity, memory illusions, and discourse processes in computer-mediated communication. He is also interested in attentional processes and cognitive representations in code comprehension and revision. His email address is: cech@usl.edu

